コードレビュー

Vol.17

CODE Review

USP 研究所技術研究員
written by 大内智明

分割したファイルを元にした、出力データファイルの作成について説明します。

連携先のデータを作成

ユニケージでは、用途に合わせてデータファイルの持ち方が変わります。例えば大量の取引データが発生するような業務では、レスポンスをよくするため、店舗単位で分割してデータを持つ場合があります。業務上、システムは他システムと連携する場合にデータ通信を行います。その際、システムごとにデータ形式（OS、データベースなど）が異なることは、よくあるので、連携先に合わせてデータ変換を行い、ファイル送信します。

今回は、自システムのデータを収集／加工／変換して、他システムに連携する方法について説明していきます。

技術的な概要

1：分割ファイルの収集

店舗毎に分割されたファイルは、まず対象リストを作成して、そのリストが1件以上存在するとデータ収集を始めます。ユニケージの場合、分割ファイルには、基本的に分割単位のコードを含むようなレイアウトとなっています。今回の場合は、店舗単位で分割しているので、店舗コードがデータのレイアウトに含まれています。今回は関係ないですが、こうして分割単位のコードを含むことで、ファイル収集後にデータ加工して、再度同じ分割コードで分割することも容易にできます（図2）。

2：全データを発注先の単位に分割して、固定長に変換

取得した全データは、連携先に I/F するデータ形式に合わせて作成します。今回は、発注先単位で I/F するため、まずは発注先単位で分割します。次に可変長のデータから固定長のデータ形式に変換します。

図1　データ作成

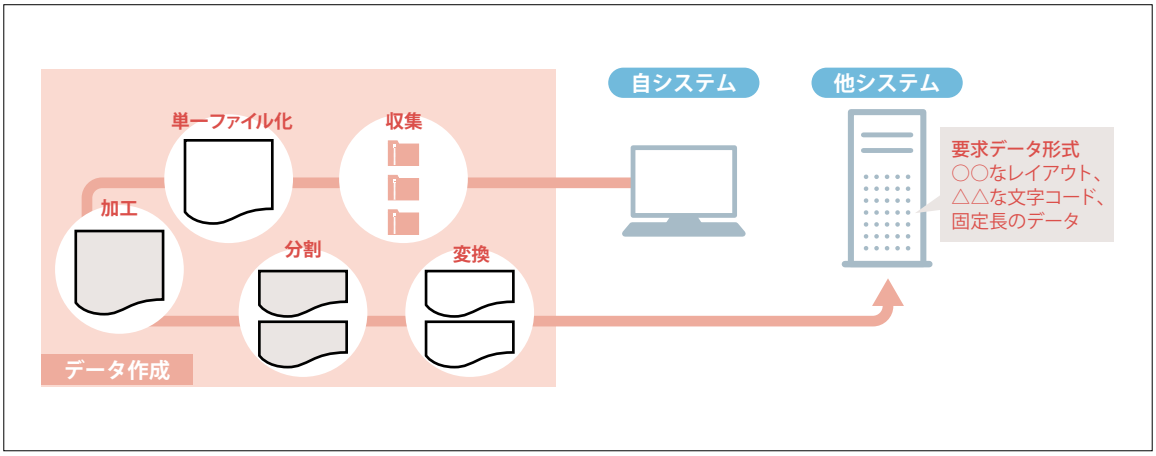


図2　ファイル収集

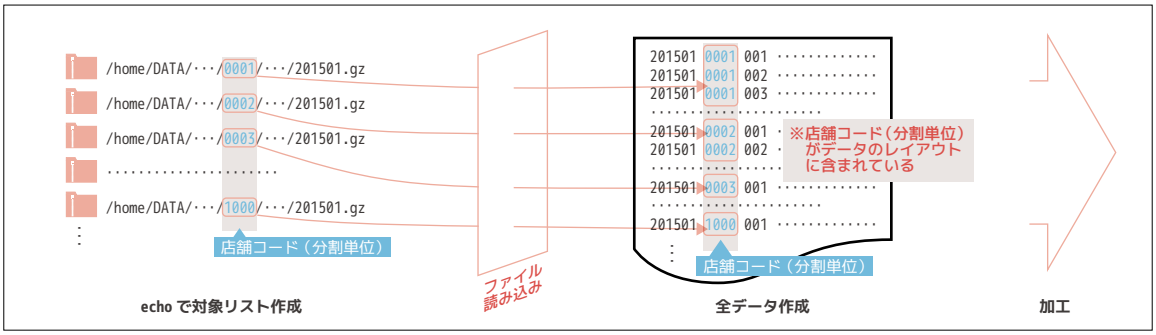
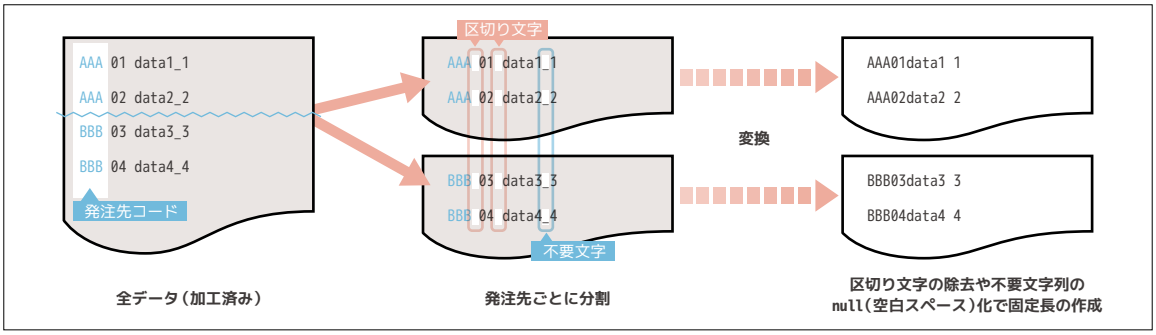


図3　ファイル分割



リスト1　複数ファイルを収集してデータ取得

```
#!/bin/ush -xve

# << 中略  >>

#------------------------------------------------------------------------------------------#
# 処理部
#------------------------------------------------------------------------------------------#

## 伝票ヘッダの参照 {{{

:> ${tmp}-header.1
for mdg in $(lineup 2 ${lv3d}/SONOTA/JGYK/TBL/JGYKCD_MDGYOMUGRP | delr 1 "000" ) ; do

#
:> ${tmp}-header.0

# 全ファイルリスト
echo ${lv3d}/DENPYOU/??????/DENPYO_HEADER_HENPIN/DYM/TBL/${nengetsu}.gz |
ugrep -v '\?'                    |
tarr                             |
while read files ; do
    [ ! -f ${files} ] && continue
    echo ${files}
done > ${tmp}-target_filelist

if [ -s ${tmp}-target_filelist ]; then
  # ファイルを100ファイルずつに
  gyocut -100 ${tmp}-target_filelist.%03d ${tmp}-target_filelist > ${tmp}-zcat_list

```

lineup コマンドはユニケージコマンド、画面1参照

指定行数でファイル分割する

```
# 2000 行ごとにファイルを分割する。
$ gyo data
5000
$ gyocut -2000 file.%02d data > list
$ cat list
file.01
file.02
file.03
$ gyo -f file.[0-9][0-9]
file.01 1000
file.02 2000
file.03 1000
```

```
30   # 100ファイル毎に解凍
31   cat ${tmp}-zcat_list                        |
32   while read files; do
33     zcat $(yarr ${files})
34     # 1:伝票年月          2:店舗コード      3:中分類コード      4:伝票種類コード          5:伝票区分コード
35     # 6:仕入先コード      7:発注先コード    8:伝票番号          9:伝票番号枝番            10:MD業務グループ
36     # 11:事業会社コード 12:伝票日付       13:訂正区分         14:EOS区分                15:企画コード
37     # 16:発注数量合計    17:数量合計       18:原価金額合計     19:売価金額合計           20:MD業務グループ
38     # 21:事業会社コード       22:店舗コード    23:原価金額合計 24:売価金額合計           25:店舗伝票入力ロック
39     # 26:本部伝票入力ロック   27:買掛月次締日 28:商管月次締日 29:伝票ステータス          30:検収入力区分
40     # 31:伝票作成区分    32:計上日         33:発注日           34:入荷予定日             35:仕入日
41     # 36:返品締日        37:返品日         38:検収日           39:移動出庫日             40:確定日
42     # 41:移動先中分類コード   42:移動伝票番号    43:売価出力区分(返品)    44:共通仕入先    45:ＥＤＩ配信コード
43     # 46:発注先ＥＤＩ配信    47:便区分     48:納品区分         49:返品区分               50:納品自動確定区分
44     # 51:発注先区分      52:センタ配信区分(発注) 53:納品EDI区分 54:センタ配信区分(返品) 55:返品EDI区分
45     # 56:発注方法区分    57:返品回数    58:物流センターコード  59:支払事業会社コード      60:卸売伝票番号
46     sleep 1
47   done                                        |
48   self 1/9 46 36 37 6 3 7 43 44 58 12 10      |
49   # 1:伝票年月        2:店舗コード          3:中分類コード        4:伝票種類コード   5:伝票区分コード
50   # 6:仕入先コード    7:発注先コード        8:伝票番号            9:伝票番号枝番     10:発注先ＥＤＩ配信コード(返品)
51   # 11:返品締日       12:返品日            13:仕入先コード       14:中分類コード    15:発注先コード
52   # 16:売価出力区分 17:共通仕入先コード 18:物流センターコード 19:伝票日付         20:MD業務グループ
53   msort key=1/9                               |
54   maezero 2.6 3.4 4.2 5.3 8.9 18.4                  > ${tmp}-header.0
55 fi
56 itouch "$(echo "_" | lcat 20 | yarr)" ${tmp}-header.0
57 ## }}} 伝票ヘッダの参照
58
59
60 # << 中略  >>
61
62 cat ${tmp}-header.0 >> ${tmp}-header.1
63
64 done
65 #
66
67 # データがなければ正常終了
68 [ ! -s ${tmp}-header.1 ] && NORMAL_END
69
70 msort key=1/9  ${tmp}-header.1 > ${tmp}-header
71
72
73 # << 中略  >>
74
75
76 ## ファイルヘッダを作成 {{{
77
78 # データ名称          | 桁数 | 説明
79 #-------------------------------------------------------------------------
80 # レコード区分        | 1    | 「A」固定。
81 # データ区分          | 2    | 「02」固定。
82 # 日付                | 6    | バッチ実行日(センター回収日)をYYMMDD形式で設定
83 # 時刻                | 6    | バッチ実行時刻をHHMMSS形式で設定
84 # データ日付          | 6    | バッチ実行日(センター回収日)をYYMMDD形式で設定
85 # 発信元コード        | 8    | 発注先コード6桁＋"00"を設定
86
87 # 固定値を作成
```

maezero コマンドはユニケージコマンド、画面2参照

ファイルの初期化を行う

```
$ cat file
cat: file: No such file or directory
$ itouch '000 000 0' file
$ cat file
000 000 0
```

ファイルが存在しないか０バイトの場合

指定文字列で初期化される

```
echo "A"                                                    > ${tmp}-kotei_base
echo "02"                                                  >> ${tmp}-kotei_base
echo "${YYMMDD}"                                           >> ${tmp}-kotei_base
echo "${HHMMSS}"                                           >> ${tmp}-kotei_base
echo "${YYMMDD}"                                           >> ${tmp}-kotei_base
echo "$(seq 1 8 | lcalc '$1*0' | yarr | tr -d " ")"       >> ${tmp}-kotei_base
yarr ${tmp}-kotei_base > ${tmp}-kotei

# ファイルヘッダは伝票ヘッダー行に対して、ひとつ作成
self 1/9 7 ${tmp}-header         |
# 1:伝票年月       2:店舗コード    3:中分類コード  4:伝票種類コード  5:伝票区分コード
# 6:仕入先コード 7:発注先コード  8:伝票番号       9:伝票番号枝番    10:発注先コード
joinx -- ${tmp}-kotei            |
#  1:伝票年月       2:店舗コード     3:中分類コード     4:伝票種類コード     5:伝票区分コード
#  6:仕入先コード   7:発注先コード   8:伝票番号         9:伝票番号枝番       10:発注先コード
# 11:レコード区分 12:データ区分     13:日付            14:時刻              15:データ日付
# 16:出荷先コード
gawk '{print $0,$7"00"}'         |
self 7 11/15 16                  |
# 1:発注先コード       2:レコード区分     3:データ区分     4:日付       5:時刻
# 6:データ日付         7:出荷先コード
msort key=1/1                    > ${tmp}-file_header_base

# 発注先ＥＤＩ配信コード(返品)毎に作成
cat ${tmp}-file_rank             |
# 1:発注先コードの識別  2:伝票番号の識別  3:伝票行番号の識別  4:発注先ＥＤＩ配信コード(返品)  5:発注先コード
# 6:伝票番号            7:伝票行番号      8:発注先ＥＤＩ配信コード(返品)
self 4 1 8 5                     |
# 1:発注先ＥＤＩ配信コード(返品)  2:発注先コードの識別  3:発注先ＥＤＩ配信コード(返品)  4:発注先コード
LANG=C sort -u                   |
msort key=4/4                    |
join1 key=4 ${tmp}-file_header_base -  |
# 1:発注先ＥＤＩ配信コード 2:発注先コードの識別 3:発注先ＥＤＩ配信コード 4:発注先コード  5:レコード区分
# 6:データ区分             7:日付               8:時刻                 9:データ日付    10:出荷先コード
msort key=1n/5n                  > ${tmp}-targert_file

sorter ${tmp}-targert_file.%1 ${tmp}-targert_file

# 発注先ＥＤＩ配信コード(返品)毎に処理
lineup 1 ${tmp}-targert_file     |
while read ht_code; do
  [ -s ${tmp}-targert_file.${ht_code} ] || continue

  # ディレクトリ作成
  mkdir -p ${output_dir}/EDI.${sday}
  # ファイル生成
  cat ${tmp}-targert_file.${ht_code}  |
  self 7/NF                           |
  # 1:日付 2:時刻  3:データ日付  4:出荷先コード
  tr -d  " "                          |
  tr "_" " "                          |
  tr -d "\n"                          > ${output_dir}/EDI.${sday}/he${ht_code}.${sdaytime}.${$}
  mv ${output_dir}/EDI.${sday}/he${ht_code}.${sdaytime}.${$} ${output_dir}/EDI.${sday}/he${ht_code}.${sdaytime}
done
```

joinxコマンドはユニケージコマンド、画面3参照

sorter コマンドはユニケージコマンド、画面4参照

画面 1　lineup

```
指定したフィールドのデータのラインナップを取り出す
$ cat data
1 1 717300 abc1
1 2 717301 abc2
1 3 717302 abc3
1 1 717303 abc4
$ lineup 1 2 data
1 1
1 2
1 3
```

画面 2　maezero

```
前ゼロを付ける
$ seq 2
1
2
$ seq 2 | maezero 1.3
001
002
```

画面 3　joinx

```
総掛けで連結する
$ cat data1
1 農業
2 工業
$ cat data2
1 東京
2 大阪
$ joinx data1 data2
1 農業 1 東京
1 農業 2 大阪
2 工業 1 東京
2 工業 2 大阪
```

画面 4　sorter

```
キーでファイル分割する
$ cat data
04 神奈川県 13 横浜市 92 56 83 96 75
01 埼玉県 03 熊谷市 82 0 23 84 10
03 千葉県 10 千葉市 52 91 44 9 0
02 東京都 04 新宿区 30 50 71 36 30
01 埼玉県 01 さいたま市 91 59 20 76 54
03 千葉県 12 柏市 95 60 35 93 76
04 神奈川県 16 小田原市 45 21 24 39 03
02 東京都 05 中央区 78 13 44 28 51
$ sorter data.%1 data
$ ls -l data.*
-rw-r--r-- 1 usp usp 87 2月 19 11:14 data.01
-rw-r--r-- 1 usp usp 82 2月 19 11:14 data.02
-rw-r--r-- 1 usp usp 77 2月 19 11:14 data.03
-rw-r--r-- 1 usp usp 91 2月 19 11:14 data.04
$ cat data.01
01 埼玉県 03 熊谷市 82 0 23 84 10
01 埼玉県 01 さいたま市 91 59 20 76 54
```

4 つのファイルに分割された

コードの見どころ

[1] 複数ファイルを収集してデータを取得します。
（❶11 行目～ 70行目まで）

[2] 取得したデータを加工して、出力用データ（フィールド形式）を作成します。
（❷88 ～ 120行目まで）

[3] フィールド形式のデータから固定長のデータを作成します。
（❸128 ～ 142行目まで）

まとめ

システム間の連携には、多種の I/F 形式が存在しています。ユーザ I/F としては、主に Web 画面、エクセル、CSV などがあります。他業務システムとの連携には、CSV、固定長レコード、マルチレイアウト、XML などがあります。

一般的に業務系システムの場合、自システムだけで完結するようなものはほとんどありません。既存のデータ形式に対応して変換することは、当然であり、今後は、新たに登場するデータ形式にも対応していく必要があります。